Sentiment Analysis

A Probabilistic Approach

March 13, 2012

Supervised by Mr. I. Langbroek (Blauw Research)
Mentored by dr. M. van Someren (Universiteit van Amsterdam)

S. A. Gieske	S. Laan	D. S. ten Velthuis	C. R. Verschoor	A. J. Wiggers
6167667	6036031	0577642	10017321	6036163

Artificial Intelligence
Faculty of Science
Universiteit van Amsterdam

Contents

1 Introduction

Due the recent growth of Social Media, such as Facebook, Twitter and blogs, consumers gained a place to review, rate and recommend products online. This online opinion is important for companies who want to market their products, manage their reputations and identify new opportunities. The process of finding relevant content, filtering out noise and categorization of messages can be automated, making it several times faster than manual processing by humans. Sentiment analysis is the application of natural language processing, computational linguistics and text analytics to identify and extract subjective information in source materials. It readies data from social media for further analysis. By performing sentiment analysis, messages can be clustered into categories (e.g. positive, negative and neutral messages).

2 Problem Specification

The supervisor, Ivo Langbroek, provided a spreadsheet of 10.000 messages from social media [1]. All messages were manually placed into 5 categories, ranging from very negative to very positive. The goal of this project is to create a method to classify these messages automatically. Before classification the contents of a message needs to be prepared. This is done by cleaning, manipulating and reducing the features inside a message. The preparation of data is described in the following sections.

2.1 Data Cleaning

A tweet can contain only 140 characters. Therefore, consumers express themselves by using slang and emoticons. It is also common to find spelling errors in the messages. This results in more noise in the data and therefore it will be important to clean the data. Punctuation usually does not add to the sentiment of the message. Therefore, all punctuation except exclamation marks and question marks are ignored. These two do give value to the sentiment of a sentence as they illustrate the meaning of the emotion used by the consumer. The emotion can also be illustrated with use of emoticons, which consists of various punctuation marks. Emoticons will be substituted for special words to maintain their contribution to the sentiment of a sentence. A Dutch stemmer, found in the Natural Language Processing Toolkit (NLTK)[2] for Python, enables reducing inflected words to their stem, the base or root form. For example the words ‘Evangelisch and ‘Evangelische are two words, which lead to the same feature as they are related to each other. By stemming ‘Evangelische into its base form ‘Evangelisch, the amount of unique words within the corpus is decreased. This increases the performance of classification as the algorithms have to process less data.

[1]Data contained posts from Blogs, Facebook, Hyves, fora and tweets
[2]http://www.nltk.org/

2.2 Data Reduction

The dataset consists mostly out of tweets. These short messages contain a single subject, whereas blog and Facebook posts are in general longer messages that may contain several opinions about several subjects. As Twitter is the largest source of information, the classification algorithms implemented are based on these type of posts. There are several methods to further decrease the vast amount of data:

- Remove words that occur only once in the corpus.
- Remove duplicate messages.
- Remove words that do not contribute to classification such as personal pronouns, stop words, and prepositions.

The efficiency of above named methods varies per machine learning method, for example, an approach that considers grammar to be an important feature for classification makes it impossible to remove words from a sentence without affecting the results.

3 Theory

3.1 Machine Learning

Machine Learning is a branch in the field of Artificial Intelligence (AI) that researches and develops algorithms capable of improving predictions or behaviours based on a dataset. An algorithm can take example data to capture the pattern in the underlying probability distribution. A successful algorithm can automatically find these complex patterns in the training data and make intelligent decisions in new similar data. The type of machine learning methods that are used in this assignment is called 'supervised learning'. This type requires that the desired prediction, in this case, the sentiment of the messages, is already known.

3.1.1 Perceptron

A perceptron is an example of a supervised learning algorithm. A perceptron is an artificial neuron, which consists of has one or more inputs s_i with corresponding weights w_i, a threshold function t and an output o. This output can be defined as $o = t(\sum_i (w_i * s_i))$. The algorithm that enables 'learning' a correct threshold takes the following number of steps:

1. Select training data from the dataset: input and the corresponding (correct) output
2. Feed the perceptron the input
3. Calculate the output
4. Compare the output to the desired output

5. Adjust weights and go to step 2 until satisfied

This algorithm adjusts the weights w_i, trying to maximize the number of correct outcomes. It should be noted that a perceptron has only a single output, making it hard to perform Multiclass Classification, when there is more than one class. For multiclass classification other algorithms can be used like the Artificial Neural Network, which is the Multiclass variant of the perceptron and described in the following section.

3.1.2 Artificial Neural Network

An Artificial Neural Network (ANN) is a computational model with the same functional aspects as biological neural networks. It consists of a group of connected artificial neurons and changes structure based on internal and external information. The neurons are similar to perceptrons, the only difference being that an output of one neuron can be the input for another. A typical network consists of an input layer, one or more hidden layers and an output layer. The number of nodes per layer can vary. The learning process is very similar to that of a perceptron but is slightly more difficult since the weights are not independent.

3.1.3 Naive Bayes Classifier

The Naive Bayes Classifier classifies a sentence using Bayes theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} P(A|B_1 \dots B_n) = \frac{P(B_1|A) \dots P(B_n)P(A)}{P(B_1 \dots B_n} \tag{1}$$

Naive Bayes is naive in the sense that it assumes all features to be independent, i.e. the occurrence of one feature is not correlated to the occurrence of another. To calculate the probability of a sentence for a certain class, the conditional probabilities of the features are needed. In order to get these probabilities for all the features, all occurrences of the features are counted. If a feature has not occurred in the training set, a count of one is used[3]. All of the conditional probabilities are multiplied together. Now the prior probability of the class is needed. This is the probability it belongs to that class when all features are ignored, i.e. the size of the class in respect to training class. A last multiplication with the prior probability gives the probability for the specified class. Note that we can forget the denominator, as they are all the same for each class.

3.1.4 Maximum Entropy

An Maximum Entropy algorithm can automatically extract a set of relationships inherent in the trainingdata and then combine these relations into a model. The idea behind Maximum Entropy models is that one should prefer the most uniform models that satify a given constraint. Max Entropy models are feature-based models. Unlike Naive Bayes, Max Entropy makes no assumptions about the relationships between features, and so

[3]This is called add-one smoothing. Other smoothing methods do exists, but were not tried.

might potentially perform better when conditional independence assumptions are not met. This means we can add features like bigrams and phrases to Max Entropy without worrying about features overlapping.

3.1.5 Weighted Probability Sum

This method is a variation on the classic Naive Bayes classifier in that it calculates the probability for a sentence to belong to a certain class by the sum of its elements, e.g. the probability that the sentence 'Ik hou van de eo' contains a positive sentiment can be defined as the sum of the probabilities of the words 'Ik', 'hou', 'van', 'de' and 'eo' to contain positive sentiment. Mathematically, this looks like: $P(s, C) = \sum_{i=0}^{n} P(s[i], C)$, where C is the class, s is the sentence and $s[i]$ is the i^{th} element of s. This approach uses each word in the sentence as a feature. It is however possible to take n words as a feature, called an n-gram. A sentence of length l where $n \leq l$ words contains $l - (n - 1)$ sequences of n words. Negations can be included for $n \geq 2$, double negations for $n \geq 3$, etc. The use of n-grams as well as 1-grams greatly improves the result of the method.

3.2 Classification Measures

In order to test the performance of the algorithms, a couple of measurements are used. Each will be explained briefly using the elements in this table **??**.

		Classified Class	
		Positive	Negative
Actual Class	Positive	TP	FN
	Negative	FP	TN

Table 1: Table of classification classes

3.2.1 Accuracy

The accuracy of a method is a measure of how much correct classifications took place $= \frac{TP+TN}{TP+FP+FN+TN}$.

3.2.2 Precision

Precision represents the measure of how many items that are classified as relevant (positive) are actually positive. It is defined as follows:

$$Precision = \frac{TP}{TP + FP}$$

3.2.3 Recall

Recall represents the measure of how many relevant items are actually classified as positive. It is defined as follows:

$$Recall = \frac{TP}{TP + FN}$$

For this assignment, recall (or precision, based on the classes) is probably the most important of the three. If sentimental sentences are seen as the positive class 1 and non-sentimental sentences as the negative class 0, the recall is based on the amount of sentimental sentences from the test set that are actually predicted as being positive class. Low recall would thus mean that the sentimental messages ‘the relevant data are seen as negative and thrown away. However, if non-sentimental messages are class 1 and sentimental message are class 0, the perspective is different.

4 Usage and User Guide

In this section the usage of the online classification is described. The online classification service should be used in the following way:

1. Send a request to www.yourwebhost.com/path/to/service/?**message**=De eo is heel cool&**dataset**=1
   Message: is the message to be classified
   Dataset: is the number of the dataset to be used.

2. Receive the resulting XML file.

5 Results and Approach

In this section the results are presented and discussed found by the various algorithms that have been implemented. This section discusses the following: the results, the global approach, binary classification and real time classification.

5.1 Naive Bayes Classifier

This section provides and discusses, the results of the Naive Bayes multi-classification algorithm. There were two implementations of the Naive Bayes Classifier, the self-made implementation and the Natural Language Tool Kit (NLTK) implementation. The implementations gave the following results:

Result table self-made implementation:

Message type	Accuracy	Precision	Recall
Positive	0.81	0.39	0.43
Negative	0.71	0.17	0.40
Neutral	0.59	0.79	0.61

Result table NLTK implementation:

Message type	Accuracy	Precision	Recall
Positive	0.80	0.34	0.42
Negative	0.72	0.18	0.40
Neutral	0.58	0.79	0.60

Conclusion
The Naive Bayes multi-classification algorithm results in a low recall and precision for the messages containing sentiment. This is the case for both implementations eventhough the self-made implementation performs better than the NLTK implementation. Therefore, it is not a suitable classification algorithm to solve the sentiment analysis problem.

5.2 Weighted Probability Sum

This section provides and discusses, the results of the Weighted Probability Sum binary-classification algorithm. The algorithm gave the following results:

Result table Neutral versus Non-Neutral:

Message type	Accuracy	Precision	Recall
Neutral	0.78	0.78	0.99
Non-Neutral	0.77	0.85	0.16

Result scatter plots Neutral (green) versus Non-Neutral (red):

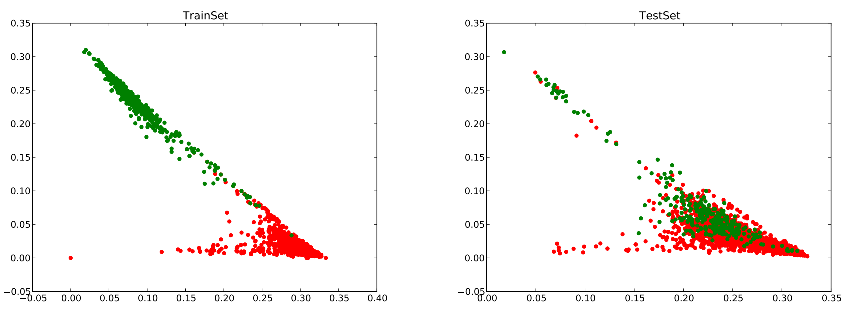


Figure 1: Scatter plots of the Training set (left) and Test set (right).

Result table Positive versus Negative:

Message type	Accuracy	Precision	Recall
Positive	0.83	0.93	0.76
Negative	0.75	0.91	0.81

Result scatter plots Positive (green) versus Negative (red):

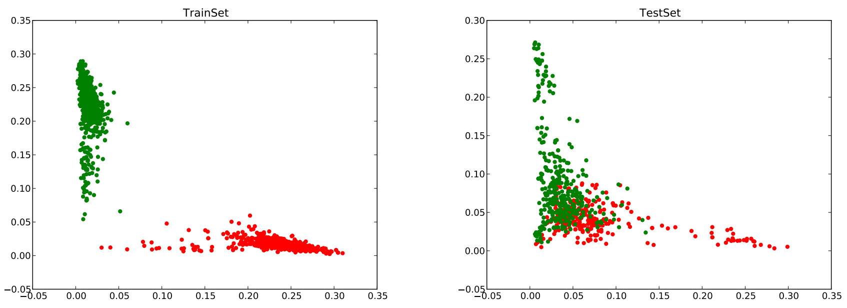


Figure 2: Scatter plots of the Training set (left) and Test set (right).

Conclusion
The weighted probability sum performs better, since the probabilities do not get too small. The probabilities of the Naive Bayes approach were so small, that new examples could not be classified correctly: it was no better than guessing. Using the weighted sum approach, the probabilities became not that small. Therefore, a distinction could still be made even with new examples. Figure **??** and **??** show the probabilities of two classes. In figure **??**, the horizontal axis is the probability the tweet is negative, the vertical axis is the positive probability. As you can see this approach is relatively good when it comes to positive versus negative classification.

5.3 Perceptron

This section provides and discusses, the results of the Perceptron binary-classification algorithm. In the Weighted Probability Sum algorithm the perceptron was already used to find a threshold. This section only discusses the perceptron with a word vector as input. The algorithm gave the following results:

Result table Neutral versus Non-Neutral:

Message type	Accuracy	Precision	Recall
Neutral	0.77	0.79	0.95
Non-Neutral	0.77	0.65	0.27

Conclusion
The Perceptron binary-classification algorithm results in either a low recall or a low precision for the messages containing sentiment. This means that a linear threshold is not good enough. Therefore, this is not a suitable classification algorithm to solve the sentiment analysis problem.

5.4 Multi-classification with a Perceptron

This section provides and discusses, the results of the Perceptron multi-classification algorithm. The algorithm gave the following results:

Result table Positive versus Negative versus Neutral:

Message type	**Accuracy**	**Precision**	**Recall**
Positive	0.77	0.65	0.26
Negative	0.65	0.62	0.31
Neutral	0.80	0.67	0.42

Conclusion
The Perceptron multi-classification algorithm results in a low recall for all classes. This means that a great amount of the messages containing sentiment are thrown away. Therefore, this is not a suitable classification algorithm to solve the sentiment analysis problem.

5.5 Neural Network

This section provides and discusses, the results of the Neural Network binary-classification algorithm. The algorithm gave the following results:

Result table Neutral versus Non-Neutral:

Message type	**Accuracy**	**Precision**	**Recall**
Neutral	0.70	0.70	0.89
Non-Neutral	0.70	0.45	0.7

Conclusion
The Neural Network binary-classification algorithm classifies many messages incorrectly. This might come due to the skewed distribution of the messages. The Neural Network will perform better in classifying, when the dataset is distributed more evenly. A larger dataset can also increase the performance of this algorithm. Therefore, this still can be suitable classification algorithm to solve the sentiment analysis problem.

5.6 Support Vector Machine

This section provides and discusses, the results of the Support Vector Machine binary-classification algorithm. The algorithm gave the following results:

Result table Neutral versus Non-Neutral:

Message type	**Accuracy**	**Precision**	**Recall**
Neutral	0.74	0.72	0.76
Non-Neutral	0.74	0.45	0.16

Conclusion
The Support Vector Machine binary-classification algorithm results in a low recall. This means that many messages containing sentiment are thrown away in the classifying proces. This means that the algorithm performs poorly overall. Therefore, this is not a suitable classification algorithm to solve the sentiment analysis problem.

5.7 Maximum Entropy

This section provides and discusses, the results of the Maximum Entropy multi-classification algorithm. The algorithm gave the following results:

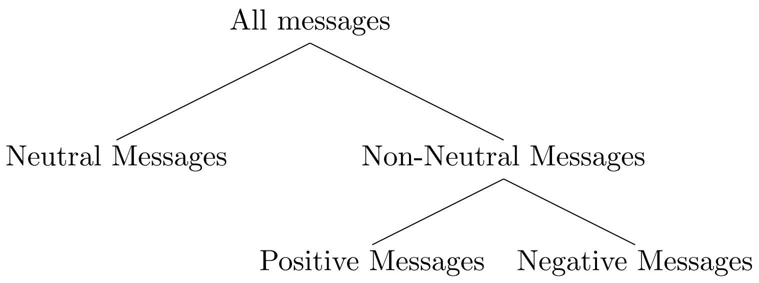
Result table Positive versus Negative versus Neutral:

Message type	**Accuracy**	**Precision**	**Recall**
Positive	0.74	0.26	0.45
Negative	0.87	0.50	0.21
Neutral	0.69	0.80	0.76

Conclusion
The Maximum Entropy multi-classification algorithm results in a low recall and precision for the positive and negative class. Therefore, this is not a suitable classification algorithm to solve the sentiment analysis problem.

5.8 Global Approach

In this section the general approach will be explained. The idea is to use a binary classifier, i.e. either true or false, multiple times. This way it is possible to first decide whether a message contains sentiment and then decide between the non-neutral messages whether it is a positive or negative message. This tree structure shows the used approach more clearly: As you can see, this approach distinguishes between two classes at a time. This is very convenient because there are a lot of techniques that can discriminate between two classes, whereas multiclass-classifiers are less common.



5.9 Binary Classification

In this section, the method that we use to classify a message is described. The approach consists of a few steps:

1. The counting of word frequencies
2. Calculation of word probabilities
3. Calculation of message probabilities
4. Finding a good threshold

First the frequency of each word, i.e. we count how many times it occurs in the training set, is counted. The number of times the word occurs in the opposite class, is equivalent to the total number of encounters minus the encounters in the first class. When the frequencies of all words[4] are found, the probability for each word can be calculated. This gives an idea of how likely it is to be encountered in a certain class. The formula for this probability is the following:

$$P(word) = \frac{\sum word \in C_1}{\sum word \in C_1 \cup C_2} \tag{2}$$

This means that the probability that a word is in C_1, the first class, is the number of times it has been encountered in a sentence, which was tagged to be in C_1, divided by the total number of encounters. Now that all words have probabilities assigned to them, or at least all words in the training set, the probabilities of the sentences can be calculated.

$$P(s) = \frac{1}{n} \sum_{w \in s} P(w) \tag{3}$$

Where s is the sentence, n the number of words in the sentence and w a word in the sentence. As follows from the formula, the weighted sum of the word probabilities is used. Now that every sentence has an associated probability, the real machine learning can begin. The goal is to find a threshold for the probabilities; when a sentence probability is higher than a certain amount, it is classified it as C_1, otherwise it belongs to C_2.

[4]All words in the training set

5.10 Real Time Classification

During the process of scraping the messages it is best to do this efficiently by classifying the message immediately. In order to do this efficiently the classification has to be real time. Therefore, saving the resulting configurations of the trained algorithms increases the performance of the classifier. Due to this the algorithms do not have to train during the classification since this is done beforehand. The saved configurations can simply be loaded into the classifier, which speeds up the efficiency of the classifying algorithm. The method currently used is the following:

1. Beforehand
   - Train the algorithms
   - Save the resulting configurations
2. Real time
   - Give the classifier input
   - Process the input
   - Load the configurations into the classifier
   - Classify input

The sources of the messages are all online, which means that its efficient to do the classification online. Therefore a online service is developed where a message can be inputted by the user. The service classifies the message in real-time and then returns the result to the user. In the current implementation the message is processed by the Hypertext Preprocessor (PHP) and Python. The incoming message is parsed by a PHP script that then calls a Python script. In the Python script the message is processed and classified. After classifying, the Python script returns the result to the PHP script, which generates a XML file containing the results. The process in short:

$$Request \rightarrow Server(PHP/Python) \rightarrow Result(XML) \tag{4}$$

6 Discussion

The several implemented methods each have their advantages and disadvantages. The most important two are the neural network and the weighted sum probability method, since these two approaches are used in the final classification of opinion vs non-opinion and positive vs negative respectively. Properties of the neural network:

- Long training time.
- Works with relatively small datasets for training.

Properties of weighted sum probability method:

- Better predictions than regular multiclass classification by using the one vs. all-approach.
- Use of n-grams takes negations into consideration.
- Needs somewhat larger datasets than neural networks.

It should also be noted that these approaches were focused on a single subject. If a more general classifier is needed, the first thing necessary is generalization of data. While domain-specific datasets such as the given one contain keywords that lead to classification (e.g. the word 'christen' often leads to negative sentiment in a sentence in this dataset, this does not have to be the case for other datasets) words such as 'leuk' or 'slecht' should be prioritized. It is, for that reason, possible to create a general-purpose sentiment analyzer, i.e subject independent sentiment classification. This may however decrease the performance, as sentences are often context dependent. To be able to cope with this problem a more sophisticated approach has to be taken. We would suggest to look more into the semantic analysis of sentences, i.e. build a tree representation of the sentence.

7 Conclusion

The manual classification process of online messages is more accurate when compared to automatic classification. The problem is that classification time is as important as accuracy, if not more, especially when thousands of messages per hour have to be processed. The methods discussed in this report are only a few approaches to the problem, with quite satisfying results. More data will almost always lead to more correct classifications, but it is also important to make correct predictions when little data is available. Neural networks may be able to do this, more so than any of the other approaches. The weighted sum method, that relies on frequencies, will perform much better when more data is involved. However, simple keywords can be used instead of the entire text to make an estimate of the sentiment in a sentence.

8 References

1. Bo Pang, Lillian Lee, *Opinion Mining and Sentiment Analysis*, Foundations and Trends in Information Retrieval. Volume 2 Issue 1-2, January 2008

2. Ainur Yessenalina, Yejin Choi, Claire Cardie, *Automatically generating annotator rationales to improve sentiment classification*, Proceedings of the ACL 2010 Conference Short Papers, pages 336-341, Uppsala, Sweden, 11-16 July 2010

3. Scot Nowson, *Scary Films Good, Scary Flights Bad. Topic driven feature selection for classification of sentiment*, In TSA '09: Proceeding of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion (2009), pp. 17-24

4. Ramanathan Narayanan, Bing Liu and Alok Choudhary, *Sentiment Analysis of Conditional Sentences*, Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 180-189, Singapore, 6-7 August 2009.